AD-A066 343 STANFORD UNIV CALIF SYSTEMS OPTIMIZATION LAB F/G 9/2
SOFTWARE FOR OPTIMIZATION.(U)
DEC 78 L NAZARETH N00014-75-C-0267
UNCLASSIFIED SOL-78-32 NL

1 OF 1
AD
AD A066343

END
DATE FILMED
5-79
DDC

Systems
Optimization
Laboratory

LEVEL II

Department of Operations Research
Stanford University
Stanford, CA 94305

SYSTEMS OPTIMIZATION LABORATORY
DEPARTMENT OF OPERATIONS RESEARCH
Stanford University
Stanford, California
94305

SOFTWARE FOR OPTIMIZATION,

by

L. Nazareth

TECHNICAL REPORT SOL 78-32
December 1978

SOL-78-32

42p.

Research and reproduction of this report were partially supported by the Office of Naval Research Contract N00014-75-C-0267; the National Science Foundation Grants MCS76-20019 A01 and ENG77-06761 A01; and the Department of Energy Contract EY-76-S-03-0326 PA #18.

Abstract

Our aim in this paper is to provide the reader with;

a) some feel for what quality software entails,

b) an overview of various aspects of optimization software,

c) information on solution techniques and available software in the form of a decision tree.

d) An extensive bibliography so that the reader can further pursue specific topics of interest.

We concentrate upon linear programming, non-linear unconstrained optimization and related areas, and non-linear programming.

This paper is intended to supplement an earlier oral presentation at the Texas Conference on Mathematical Software entitled "State of Software for Optimization".

Acknowledgment

My grateful appreciation to Professor G.B. Dantzig and the many others who helped make my visit to the Systems Optimization Laboratory an interesting and valuable learning experience.

My thanks also to the Applied Mathematics Division, Argonne National Laboratory, who jointly with the Systems Optimization Laboratory supported my appointment at Stanford.

SOFTWARE FOR OPTIMIZATION

by

L. Nazareth

1. Introduction

This paper is intended to supplement an earlier oral presentation on developments in optimization software.* Since very many mathematical problems can be posed in terms of function optimization, and since software for each optimization area ranges from small scale pilot programs to the large scale systems of which commercial LP systems are the most familiar example, we are of necessity selective in our choice of subject matter. We concentrate upon the areas of linear programming, non-linear unconstrained optimization (and the related areas of non-linear least squares and systems of non-linear equations), and non-linear programming. In particular, the very important areas of discrete variable programming and dynamic programming are not covered here.

Our aim in this paper is to provide the interested reader with:

a) Some feel for what quality software entails.

b) An overview of various aspects of optimization software.

---

*"State of Software for Optimization" -- invited presentation at Texas Conference on Mathematical Software, April 1978.

c) Information on solution techniques and available software for the above areas in the form of a decision tree.

d) An extensive bibliography, so that the reader can further pursue specific topics of interest.

The paper is organized as follows: Section 2 provides some historical background to the optimization areas covered. Section 3 gives an overview of the software development process, and discusses attributes of 'quality' mathematical software, illustrating these with specific examples. Section 4 deals with software primarily intended to aid algorithm and code development, and discuss the idea of a language for mathematical programming. Section 5 deals with available optimization software for the areas covered here. A detailed decision tree is given. Section 6 discusses the testing of software and the bibliography is given in the final section.

We make no claims to being complete, but the author would welcome feedback on important omissions and inaccuracies, particularly with regard to material in Section 5.

2. Background

Table 1 is designed to give the reader a time frame for developments in optimization. Some of the important theoretical and algorithmic references are listed along with a few parallel developments in software and computers, and the reader can superimpose his own set of favorite topics.

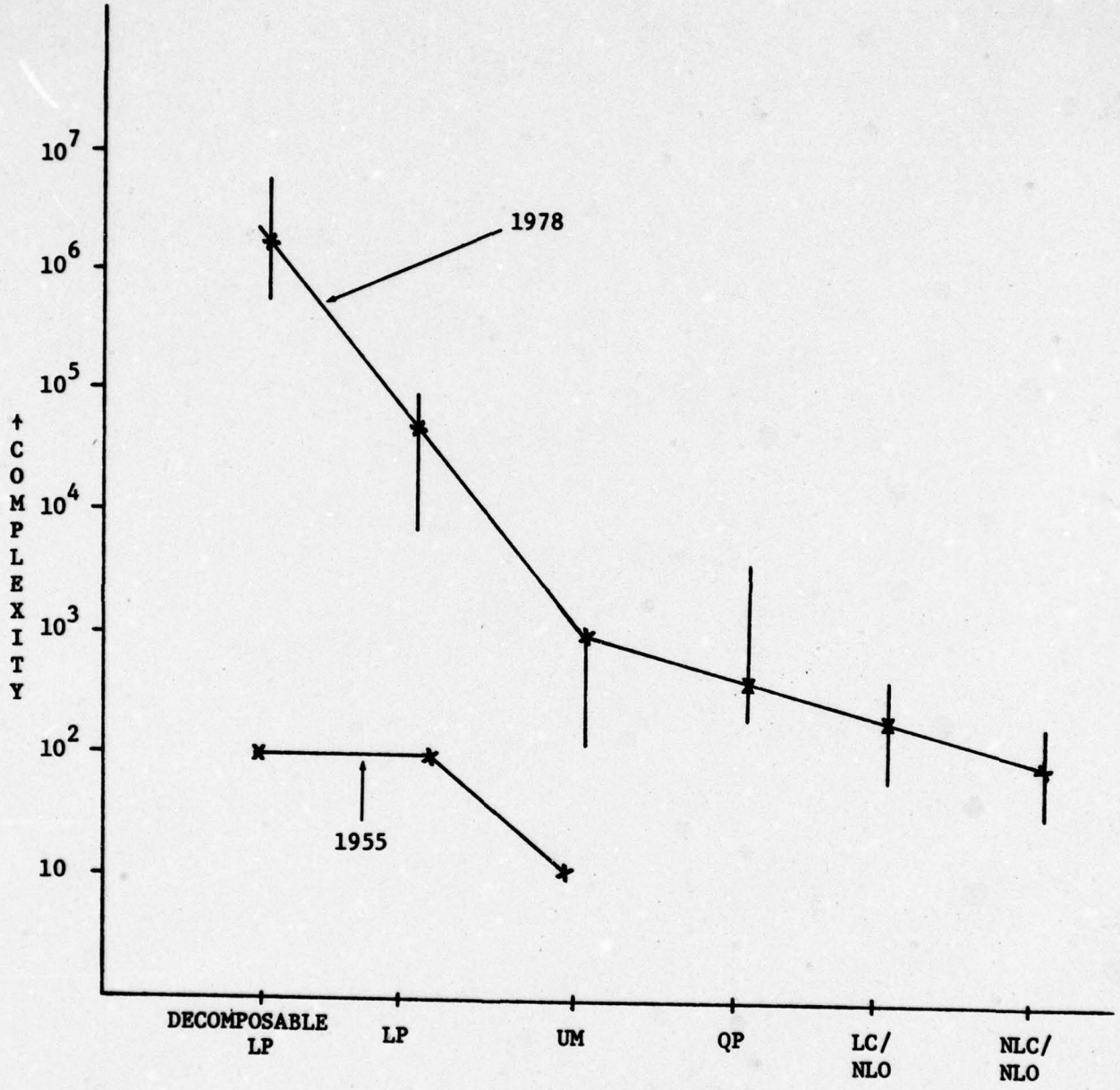
Table 2, adapted from Wolfe [1975b], shows how our ability to solve problems has increased substantially. The vertical axis represents complexity of the problem and the horizontal axes lists different areas of optimization.

For further historical details see the survey articles of Dantzig [1977], Orchard-Hays [1977] and Wolfe [1975b].

Table 1

DATES	ALGORITHMS & THEORY	SOFTWARE	COMPUTERS
†	Newton (1727); Fourier (1823);		
1935	Babbage (1840)		
	Motzkin (1936)-Inequality Theory;		
1940	Kantorovitch (1939)-L.P.		
1945	VonNeumann (1944)-Game Theory		
	Dantzig (1947)-L.P., Simplex Method		First Generation
1950			ENIAC, UNIVAC, EDVAC, ACE, EDSAC
	Orchard-Hays (1952/4)-Code for Simplex Method; Hestenes & Steifel (1952)-	EDSAC Library	
	Conjugate Gradients; Ford & Fulkerson	LP 10 rows (1952)	SEAC at N.B.S.
1955	(1954)-Network Flows	LP 100 rows (1954)	IBM 701
	Gomory (1958)-Integer Programming;	LP 256 rows (1956)	IBM 704
	Wolfe (1959)-Q.P.	First matrix generator (1956)	
1960	Davidon (1959)-Variable Metric	LP 90/94-1000 rows	Second Generation
	Dantzig-Wolfe (1960)-Decomposition; Rosen (1960)-Gradient Programming; Wilkinson (1960)-Error Analysis; Wilson (1963)-Constrained Optimization;		IBM 7090, CDC 3600
1965	Fletcher Reeves (1964)-Conjugate gradients		Third Generation
	Fiacco & McCormick (1964/6)-Penalty	SUMT	IBM 360, CDC 6400
	Abadie & Carpentier (1969)-GRG; Murray (1969)-Constrained Optimization; Hestenes		
	(1969), Powell (1969), Rockafeller (1973)-	MPS/360, GRG	
	Augmented Lagrangian; Bartels & Golub	NATS Project-EISPACK	
1970	(1969)- factorizations	NAG Project, IMSL	Parallel Computers
	Powell (1971)-Convergence Analysis; Gill, Golub, Murray & Saunders (1974)-		
1975	Factorizations	MPSX/370	
	Han (1975), Powell (1977)-Variable Metric for Constrained Optimization	NPL Optimization Library	
		MINPACK (Argonne)	
		MINOS CODE (Murtagh & Saunders (1977))	
		NL2SOL (Dennis et al. (1977))	

Table 2



U.M. — Unconstrained minimization, QP-Quadratic Programming
LC/NLO -- Linear constraints, non-linear objective, NLC/NLO-non linear constraints and objectives
Complexity - (number of rows + number of variables)
‡ -- indicates the figure given is only a rough approximation between the bounds indicated

3. Quality Software

3.1. Overview of Software Development

The design of an item of mathematical software depends very much upon the intended use of the software. We wish to stress the distinction between implementations of an algorithm designed primarily for studying the behavior of an algorithm, and implementations designed primarily for solving problems. The former are called algorithm/code oriented versions and the latter user/problem oriented versions. The distinction is, of course, not clear cut, since algorithm/code oriented versions can and should be used to solve practical problems, and user/problem oriented versions can and should be used to study the encoded algorithm. However an implementation will usually place emphasis on one of these two goals; and often an algorithm/code oriented version will be developed as a prelude to a user/problem oriented version.

An algorithm/code oriented version should not be construed to mean a hastily thrown together version. Rather it indicates a version in which emphasis is placed upon the goals of flexibility, generality and modifiability, even if this results in a sacrifice of efficiency. In a user/problem oriented version, increasing emphasis is placed upon efficiently solving a wide class of problems and providing a wide range of options. This may call for substantial reformulation and reorganization of the calculations to reduce overhead and circumvent numerical difficulties.

We feel that insufficient attention has been paid to developing tools to aid the implementation of algorithm/code oriented versions and this has contributed to the proliferation of untested algorithms which abound in the literature. This will be discussed further in Section 4. In contrast, a number of software aids have been developed to aid the process of tailoring a code to a particular compiler/machine configuration, i.e., to develop portable versions. For further details see Boyle [1976]. Since such aids can be applied to most items of mathematical software and are not specialized to optimization, we shall not discuss them further here.

For a more detailed discussion of the process of mathematical software development see Nazareth [1978a].

3.2. Attributes of Quality Software

Both algorithm/code oriented and user/problem oriented software should meet certain standards. What is it that characterizes 'quality' software?

Recent efforts to develop good mathematical software, Rice [1971], Smith et al. [1974], Ford and Hague [1974], identify several attributes. We quote these and illustrate them with specific examples.

(a) Robustness refers to the ability of a computer program to detect and gracefully recover from abnormal situations without unnecessary interruption of the computer run. In situations when a calculation

does fail, the code should fail gracefully. Robustness involves, for example, the filtering out of improper arguments, the avoidance of destructive overflows, and the reorganization of a calculation to minimize the effect of rounding error.

Example 2: Cody [1976], Avoiding both destructive overflows and non-destructive underflows in the computation of $\|x\| = [\sum_{i=1}^{n} x_i^2]^{1/2}$.

The usual FORTRAN calculation proceeds as follows:

```
      SUM = 0.0 DO
      DO 10 I = 1, N
         SUM = SUM + X(I)**2
   10 CONTINUE
      XNORM = DSQRT(SUM)
```

SUM can overflow even though XNORM may be a machine representable member. In order to avoid this, the calculation can be done as follows, where we assume for convenience, that the largest element, in absolute value, is X(1).

```
      SUM = 1.0 DO
      DO 10 I = 2, N
         A = X(I)/X(1)
         SUM = SUM + A*A
   10 CONTINUE
      XNORM = DABS(X(I))*DSQRT(SUM)
```

Now x(I)/x(1) can underflow (non-destructively) leading to troublesome interrupt messages. To avoid this the computation can be further reorganized as follows:

```
      SUM = 1.0 DO
      B = DABS(X(1))
      DO 10 I = 2, N
         A = 0.0 DO
         IF  (B + DABS(X(I)).NE.B)A = X(I)/X(1)
         SUM = SUM + A*A
   10 CONTINUE
      XNORM = DABS(X(1))*DSQRT(SUM) .
```

Example 2: Reorganizing calculation to minimize effect of rounding error. When variable metric methods were first suggested for solving the problem $\min_{x \in \mathbb{R}^n} f(x)$, the calculation was stated in terms of updating an approximation to the inverse Hessian H of f(x). Given a step $\Delta x \triangleq x^* - x$ and the associated change of gradient of f(x), $\Delta g \triangleq \Delta f(x^*) - \Delta f(x) \triangleq g^* - g$, a new approximation H^* is developed, for example, by the BFGS (see Broyden [1970]) update

$$H^* = H + \frac{1}{\Delta x^T \Delta g}[\rho\ \Delta x\ \Delta x^T - \Delta x\ \Delta g^T\ H - H\Delta\ g\ \Delta x^T] \tag{3.1}$$

where

$$\rho \triangleq 1 + \left[\frac{\Delta g^T\ H\ \Delta g}{\Delta x^T \Delta g}\right] \tag{3.2}$$

In theory $H > 0$ (i.e., positive definite) $\Rightarrow H^* > 0$ whenever $\Delta g^T \Delta x > 0$. However rounding error in the computation of H^* can destroy this property. Another difficulty is that even when $H^* > 0$ but ill-conditioned, rounding error in computing the next direction of search $d^* \triangleq -H^* g^*$ can result in $d_c^* g^* > 0$, where $d_c^* = -f\ell(H^* g^*)$ is the computed search direction. See Gill, Murray and Pitfield [1972].

The above difficulties can be circumvented by reorganizing the calculation following the suggestions of Gill and Murray [1972]. They suggest working with an approximation to the Hessian B which is maintained in the factored form $B = LDL^T$, where L is lower triangular and D diagonal. In this case we can ensure positive definiteness by keeping $D > 0$. In addition a bound on the condition number of B can be improved by modifying D. See also the Example 2 under Reliability.

(b) Reliability refers to the ability of an item of software to perform a calculation both efficiently and accurately and to reflect the basic characteristics of the algorithm e.g., its scale invariancies.

Example 1: Estimating gradients of $f(x)$ by finite differences. In theory each component g_j can be estimated by first order finite differences

$$g_j = [f(x + h\, e_j) - f(x)]/h \tag{3.3}$$

where h is an infinitesmal step.

In finite precision arithmetic however this is a difficult computation. A good routine must be designed with considerable care and we state some of the issues which arise in designing such a routine.

-- Choice of step length, h.

- Should h vary with each component?
- Should h be chosen to balance rounding and truncation?

In this latter case estimates of second derivatives are needed to estimate truncation error. How are these obtained? Should h be estimated at every iteration or should it be only periodically recomputed in a separate subroutine and held fixed in between calls to this subroutine.

-- Should a switch to central differences be made when forward differences are insufficiently accurate?

-- Should the increment h be relative to $|x_j|$ or should it be absolute? In the former case g_j is invariant under a simple scaling of variables $x_j \to \alpha\, x_j$, whilst in the latter case g_j is invariant under a translation of variables $x \to x + c$.

-- Should gradients be estimated in a transformed space of variables? i.e., consider the function

$$f(z) = f(x) + g^T(z - x) + \frac{1}{2}(z - x)^T B(z - x)$$

where $B = (JJ^T)^{-1}$ and non-singular.

Contours of $f(z)$ are illustrated in Figure 3.

If we make the transformation $z = x + Jy$, then $f(z)$ transforms to $\bar{f}(y)$

$$\bar{f}(y) = f(x) + k^T y + \frac{1}{2} y^T y$$

where $k = J^T g$. The contours of $\bar{f}(y)$ are illustrated in Figure 4. k_i are estimated by

$$k_i = \frac{f(x + J_i h) - f(x)}{h} - \frac{h}{2}$$

and g is then given by $g = (J^T)^{-1} k$.

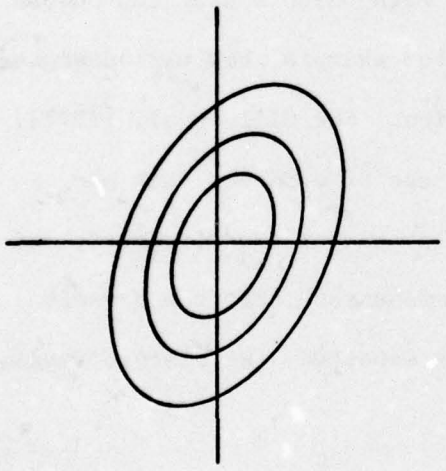


Figure 3

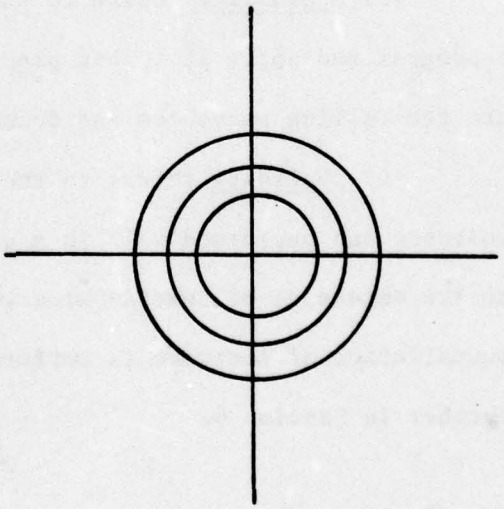


Figure 4

Example 2: Invariance w.r.t. transformations of variables. It is well known that apart from the initial choice of the approximation, the variable metric algorithm is invariant w.r.t. transformation of the variables. However, if the algorithm is modified to ensure that the search direction $d_k = -H_k g_k$ satisfies

$$|d_k^T g_k| \geq \varepsilon \|g_k\| \|d_k\| , \qquad \text{for } \varepsilon \text{ a small constant,} \tag{3.4}$$

(for example by modifying H_k suitably), then this destroys scale invariance, since (3.4) is not invariant. For a fuller discussion, see Powell [1976b].

(c) Structured refers to whether the program is designed along the principles of good programming, i.e., whether it has a top to bottom flow of control, is formatted to display its structure and so on. See Dahl et al. [1972] and Kerninger and Plauger [1974].

(d) Usability refers to the ease with which a user can choose a program and apply it to his problem. For example, how well designed are the calling sequences and documentation. See Gill et al. [1977].

(e) Validity refers to the existence of evidence that the software has performed well in a particular computer environment, and to the existence of testing aids which demonstrate that the present installation of software is performing as expected. We discuss testing further in Section 6.

(f) Transportability refers to whether an item of software can be moved from one computer installation to another without degredation of performance and with minimal change.

Example: Features of COMMON statement in FORTRAN which can hinder transportability. A very complete discussion of difficulties which arise in transporting FORTRAN programs is given in Smith [1976]. When using the COMMON statement some of the difficulties which arise are:

-- The order of variables affects portability, e.g., some IBM machines require that variables in a COMMON statement which use two storage units, begin on an even word boundary, else alignment errors or a degredation in efficiency can result.
-- Variables in labelled COMMON may become undefined upon execution of a RETURN (or END) statement, unless there is a COMMON statement for that block, in at least one of the higher level program units in the chain of active programs.
-- Other inconveniences associated with COMMON are that variable dimensioned entities cannot be used in COMMON, and that the size of a labelled COMMON block may be required to be the same in each program unit in which the COMMON block occurs.

4. Software Designed to aid Algorithm/Code Development

Implementing an optimization algorithm is a difficult and a time consuming task. For this reason it is essential that the algorithm or software developer be provided with suitable tools which facilitate his task. We can distinguish three approaches:

Approach 1. Develop a high level language, for example along the lines of the Mathematical Programming Language (MPL) of Dantzig et al. [1970]. The aim is to design a language in which highly readable programs can be written and which parallels the venacular of applied mathematics. This would make it possible to write programs quickly and easily and would serve as a means of communicating ideas precisely. In the early creative stages of algorithm development such a language is an invaluable aid since one's intuition can now be supplemented by hard computational results. This can then in turn lead to new ideas. Such a language is also a valuable educational aid. However, once the main features of an algorithm have been laid out, a fundamental difficulty remains, namely that numerically sound procedures are difficult to write in any language, no matter how convenient. What is then needed is a good library of procedures, tailored to optimization, from which an optimization algorithm can be built. This is particularly useful when one wants to test out a new algorithm on real life problems. Building optimization algorithms from a library of procedures also makes for a more uniform comparison of algorithms since their implementations can be made to differ only

in the essentials and test results are thus less subject to variations in programming style. It is difficult for a new language to gain wide acceptance and for compilers to be made available on a wide range of machines. Often therefore, we have to fall back upon FORTRAN, although other high level languages, e.g., PL-1 and ALGOL-68 are making some headway.

Approach 2. Use individual components of a user/problem oriented implementation (or optimization system) which has a modular design. Examples in the area of Linear Programming are discussed in Nazareth [1978]. For examples of such systems in the area of non-linear programming see Muralidharan and Jain [1975], Hillstrom [1976]. The main difficulty with this approach is that one has usually very limited flexibility. Each component in a user/problem oriented system is usually designed within the context of the overall system and often utilizes a common data structure. Using a component on a stand alone basis usually requires that it be substantially modified.

Approach 3. The idea behind the third approach has already been mentioned. Here one seeks to develop a carefully specified set of modules which can be viewed as being the 'primitives' or 'basic operators' of a language for building optimization algorithms. Two efforts along these lines are discussed in Nazareth [1977] and [1978]. The former describes a pilot system based upon a set of algorithms developed by the author in the area of non-linear unconstrained optimization. Building upon this experience, a software organization and

development effort was undertaken in the area of Linear Programming, as described in Nazareth [1978]. It is important to emphasize that the development of a code requires careful craftmanship and should not be viewed as the mere stringing together of modules. However if such modules are carefully designed and correctly implemented, they can greatly ease the task of implementing an algorithm and perhaps they should be viewed as a way of developing an "artists sketch" of a code, which can then be further refined. They also serve as a valuable means of cooperation and communication between different researchers. Finally they are a useful educational aid.

5. Optimization Software

In this section we give a detailed decision tree of the major categories of optimization methods together with references to some recent implementations and/or algorithms in each category. References are given to journal articles or technical reports and implementations are identified by a symbol indicating their source.

Available software varies widely in quality and we do not set out here to make any value judgements. Clearly such a compact presentation is also far from complete. Our more modest aim is to provide the reader with some selected information on individual items of software. Other surveys e.g., Dennis [1976], Dixon [1973], Fletcher [1976], Wolfe [1975a], Wright [1978] should also be consulted.

5.1. Some Major Sources of Optimization Software (Alphabetical)

[a] Argonne National Laboratory, Applied Mathematics Division([hp]-Hillstrom's Package [1976] and [m]-MINPACK-1).

[b] Bell Telephone Laboratory, Murray Hill, New Jersey, (PORT Library).

[h] Atomic Energy Research Establishment (A.E.R.E.) Computer Science and Systems Division, Harwell, England.

[ibm] IBM Mathematical Subroutine Library (SL-MATH).

[imsl] International Mathematical and Statistical Libraries, Inc.

[w] Computer Center, University of Wisconsin, Madison.

[nag] Numerical Algorithms Group (NAG) Library.

[nber] National Bureau of Economic Research, Cambridge, Massachusetts (now part of M.I.T.).

[noc] Numerical Optimization Center, Hatfield College of Technology, Hatfield, Herts., England.

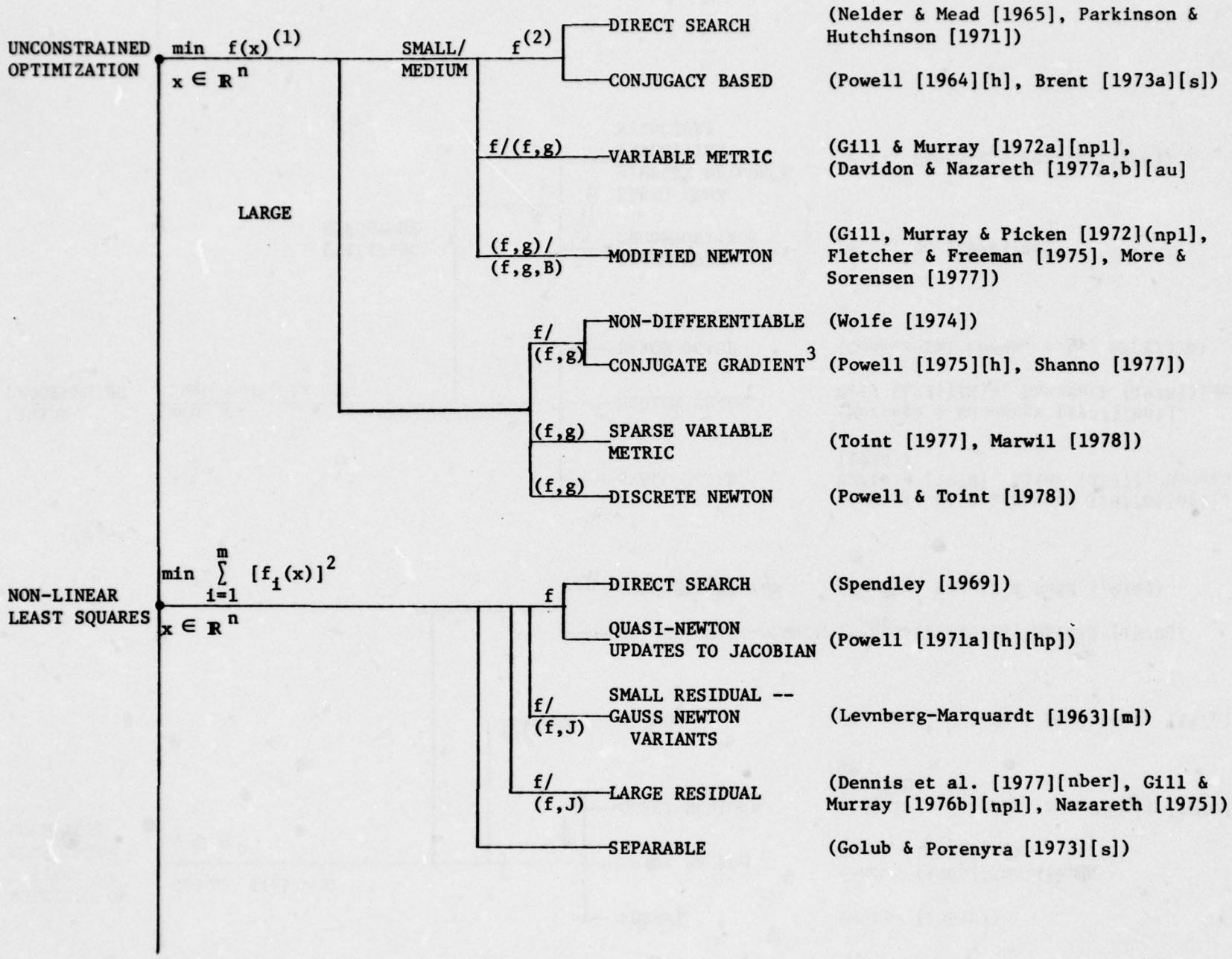
[npl] National Physical Laboratory, Division of Numerical Analysis and Computing, Teddington, England. (The NPL Optimization Library is the most comprehensive collection of optimization software currently available.)

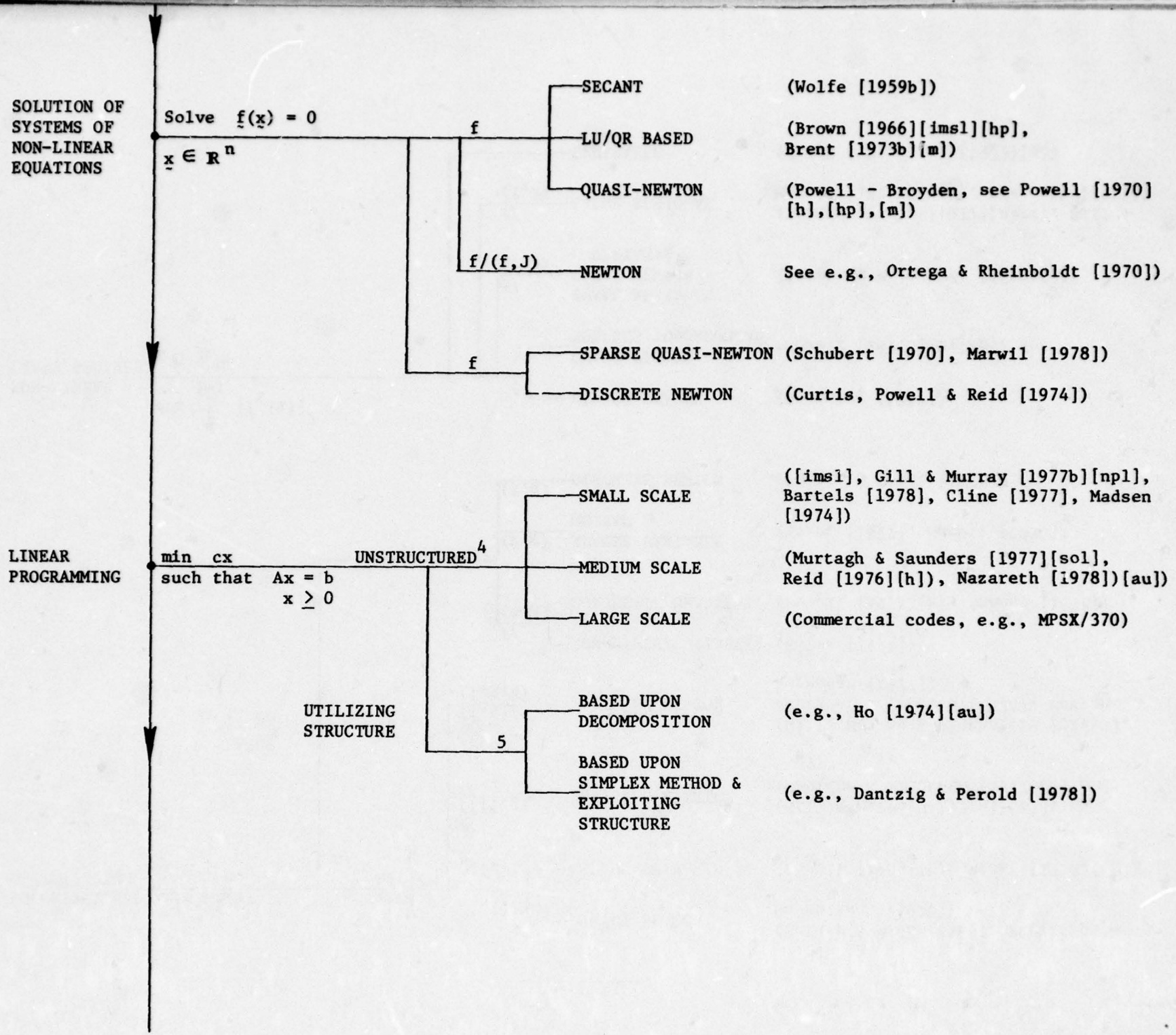
[s] Computer Science Department, Stanford University.

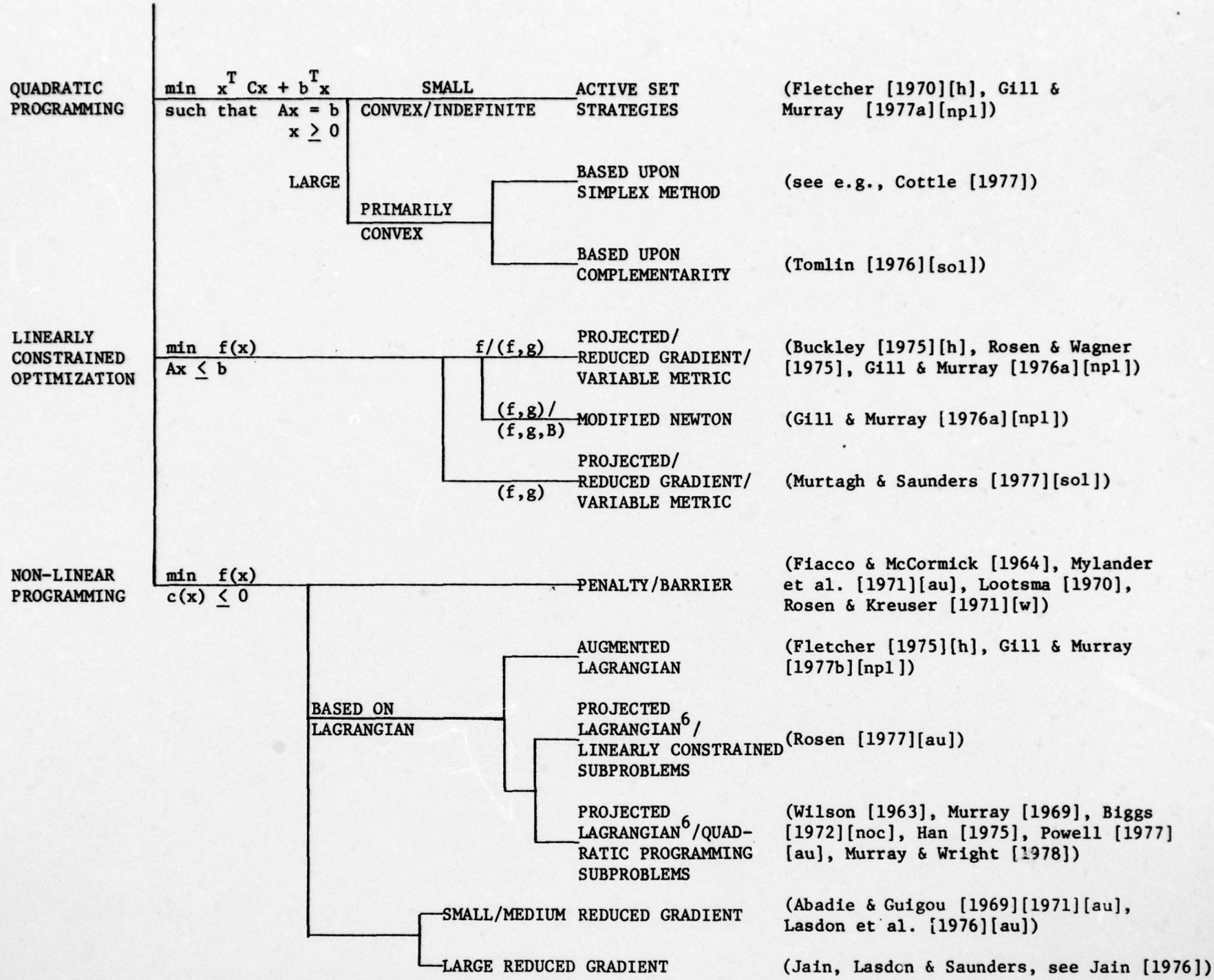
[sol] Systems Optimization Laboratory, Department of Operations Research, Stanford University.

The symbol associated with each establishment in the above list is used to identify the establishment in the Decision Tree. Note that software referred to in this manner is not necessarily available for general distribution. Conversely when an algorithm in the Decision Tree does not have a symbol associated with it or when the symbol [au] is used, an implementation may be available from the author(s) of the cited reference.

Decision Tree




SOLUTION OF SYSTEMS OF NON-LINEAR EQUATIONS
Solve f(x) = 0
x ∈ R^n
f
SECANT
(Wolfe [1959b])
LU/QR BASED
(Brown [1966][imsl][hp], Brent [1973b][m])
QUASI-NEWTON
(Powell - Broyden, see Powell [1970] [h],[hp],[m])
f/(f,J)
NEWTON
See e.g., Ortega & Rheinboldt [1970])
f
SPARSE QUASI-NEWTON
(Schubert [1970], Marwil [1978])
DISCRETE NEWTON
(Curtis, Powell & Reid [1974])
LINEAR PROGRAMMING
min cx
such that Ax = b
x ≥ 0
UNSTRUCTURED 4
SMALL SCALE
([imsl], Gill & Murray [1977b][npl], Bartels [1978], Cline [1977], Madsen [1974])
MEDIUM SCALE
(Murtagh & Saunders [1977][sol], Reid [1976][h]), Nazareth [1978])[au])
LARGE SCALE
(Commercial codes, e.g., MPSX/370)
UTILIZING STRUCTURE
5
BASED UPON DECOMPOSITION
(e.g., Ho [1974][au])
BASED UPON SIMPLEX METHOD & EXPLOITING STRUCTURE
(e.g., Dantzig & Perold [1978])


QUADRATIC PROGRAMMING
min $x^T Cx + b^T x$
such that $Ax = b$
$x \geq 0$
SMALL
CONVEX/INDEFINITE
ACTIVE SET STRATEGIES
(Fletcher [1970][h], Gill & Murray [1977a][npl])
LARGE
PRIMARILY CONVEX
BASED UPON SIMPLEX METHOD
(see e.g., Cottle [1977])
BASED UPON COMPLEMENTARITY
(Tomlin [1976][sol])
LINEARLY CONSTRAINED OPTIMIZATION
min f(x)
$Ax \leq b$
f/(f,g)
PROJECTED/ REDUCED GRADIENT/ VARIABLE METRIC
(Buckley [1975][h], Rosen & Wagner [1975], Gill & Murray [1976a][npl])
(f,g)/ (f,g,B)
MODIFIED NEWTON
(Gill & Murray [1976a][npl])
(f,g)
PROJECTED/ REDUCED GRADIENT/ VARIABLE METRIC
(Murtagh & Saunders [1977][sol])
NON-LINEAR PROGRAMMING
min f(x)
$c(x) \leq 0$
PENALTY/BARRIER
(Fiacco & McCormick [1964], Mylander et al. [1971][au], Lootsma [1970], Rosen & Kreuser [1971][w])
BASED ON LAGRANGIAN
AUGMENTED LAGRANGIAN
(Fletcher [1975][h], Gill & Murray [1977b][npl])
PROJECTED LAGRANGIAN[6]/ LINEARLY CONSTRAINED SUBPROBLEMS
(Rosen [1977][au])
PROJECTED LAGRANGIAN[6]/QUADRATIC PROGRAMMING SUBPROBLEMS
(Wilson [1963], Murray [1969], Biggs [1972][noc], Han [1975], Powell [1977][au], Murray & Wright [1978])
SMALL/MEDIUM REDUCED GRADIENT
(Abadie & Guigou [1969][1971][au], Lasdon et al. [1976][au])
LARGE REDUCED GRADIENT
(Jain, Lasdon & Saunders, see Jain [1976])

Footnotes to Decision Tree

1 A particular case of this problem is that of 1 - D optimization, both stand alone and for use within n-dimensional optimization routines. See e.g., Brent [1973a], Gill and Murray [1974].

2 The symbols f, g, B and J are used to indicate the type of information about the function that is usually required when

f stands for function value

g stands for gradient

B stands for Hessian

J stands for Jacobian.

Thus e.g., f/(f,g) means function value or (function value and gradient).

3 Currently a very active research area. For an overview see Nazareth and Nocedal [1978].

4 The distinction between small, medium and large scale is as follows. In small scale L.P. no account is taken of the sparsity of the L.P. matrix, i.e., it is assumed that the matrix is dense and is usually stored as a 2 - D array. In medium scale L.P. it is assumed that the L.P. matrix will fit in core provided only non-zeros are stored in packed form, e.g., as a column list/row index data structure. Finally, large scale systems e.g., MPSX/370 make extensive use of secondary storage.

5 For a good overview see Dantzig [1968], Geoffrion [1970].

6 Terminology of Murray and Wright [1978].

6. Testing

Evaluating optimization routines is a difficult task, and one which requires both qualitative and quantitative measures of performance. A fundamental requirement is that the testing environment simulate an actual environment of use since, if it did not, the evaluation would be valid but in all likelihood, irrelevant. Furthermore, the overall quality of a code can only be gauged after investigating a broad range of issues, for example, efficiency, robustness, usability, usefulness of documentation, ability of fail gracefully in the presence of user abuse, rounding error difficulties or violation of underlying assumptions. A testing method usually concentrates on efficiency and robustness, evaluating these by exercising the code on a set of well chosen and hopefully realistic problems.

To date the most common method of evaluating optimization routines has become known as 'battery' or 'simulation' testing.

Comprehensive studies along these lines are described in Colville [1968], Hillstrom [1977], Himmelblau [1972]. Battery testing has two basic components, namely a set of test problems and a set of measures of performance. The approach is subject to limitations which sometimes make a clear ranking of methods difficult to discern. For example, it is difficult to know how much confidence should be attached to a particular measure of performance when slight variation of starting point or geometry of test problem leads to a substantial variation in the measure of performance. For a discussion of these difficulties see Nazareth and Schlick [1976]. This has motivated

the approach which employs "problem families" or "parameterialized test problems" introduced originally into the evaluation of routines for numerical quadrature by Lyness and Kaganove [1976]. See also Dembo and Mulvey [1976]. A careful a priori experimental design and the use of statistical sampling theory and analysis are implicit in this approach, which is sometimes referred to as 'performance profile' testing to differentiate it from 'battery' testing.

A second distinction which it is worth emphasizing is the distinction between algorithm and software evaluation. In particular testing an algorithm usually places most emphasis on efficiency whilst software evaluation attaches a great deal of importance to reliability and robustness.

Finally it is worthwhile making a distinction between decentralized and centralized testing of routines. The former is illustrated by the original study of Colville [1968], and the latter is illustrated by the study of Hillstrom [1977]. In decentralized testing a set of software tools are usually made available to developers of routines who then use them to develop information about how well their routines perform. For an example in the area of non-linear programming see Nazareth [1977 where the testing tools comprise:

(i) Subroutines which return function and/or gradient information for a set of different test functions.

(ii) Subroutines which return starting point and expected solution (if known), for each function.

(iii) Report writer whose features include:

-- Flexible and convenient way of specifying which functions to test.

-- Replaceable section of code for routine being tested.

-- Interface subroutines between user form of function call and subroutines in (i) above.

-- Output summaries and graphical display.

Systems Optimization Laboratories (see Dantzig et al. [1973]) are a natural environment for centralized testing, i.e., gathering and testing a number of routines at one particular site. This usually requires a substantial commitment of resources, but it makes for a much more uniform comparison and permits the use of much more stringent test problems, in particular problems arising from real life applications (see Dantzig and Parikh [1977]).

Until fairly recently, the development of a testing methodology for optimization routines has been sorely neglected. However, the crucial importance of the subject is now being recognized. For a description of some recent work see Bus [1977], Crowder, Dembo and Mulvey [1977], Nash [1975], More et al. [1978], and consult the minutes of meetings of the Committee on Algorithms of the Mathematical Programming Society.

Bibliography

Abadie, J. and Carpentier, J. (1969), "Generalization of the Wolfe Reduced Gradient Method to the Case of Non-linear Constraints," in Optimization, R. Fletcher, (ed.), pp. 34-39, Academic Press, London and New York.

Bartels, R.H. (1978), "A Penalty Linear Programming Method Using Reduced Gradient/Basis Exchange Techniques," Johns Hopkins University, Math. Sci. Dept., Report No.

Bartels, R.H. and Golub, G.H. (1969), "The Simplex Method of Linear Programming Using LU Decomposition," Comm. ACM 12, pp. 266-268.

Biggs, M.C. (1972), "Constrained Minimization Using Recursive Equality Quadratic Programming," in Numerical Methods for Non-linear Optimization, F.A. Lootsma, (ed.), pp. 411-428, Academic Press, London and New York.

Boyle, J.M. (1976), "Mathematical Software Transportability Systems -- Have the Variations a Theme?" In Portability of Numerical Software, W. Cowell, (ed.), pp. 305-360, Springer-Verlag Lecture Notes in Computer Science, Berlin-Heidelberg-New York.

Brent, R.P. (1973a), Algorithms for Minimization Without Derivatives, Prentice-Hall, Englewood Cliffs, New Jersey.

Brent, R.P. (1973b), "Some Efficient Algorithms for Solving Systems of Non-linear Equations," SIAM J. Numer. Anal. 10, pp. 327-344.

Brown, K.M. (1966), "A Quadratically Convergent Method for Solving Simultaneous Nonlinear Equations," Purdue University, Ph.D. Dissertation, Lafayette, Indiana.

Broyden, C.G. (1970), "The Convergence of a Class of Double-rank Minimization Algorithms, Part 1 and 2," J. Inst. Math. Applics. 6, 76-90, pp. 222-231.

Buckley, A.G. (1975), "An Alternate Implementation of Goldfarb's Minimization Algorithm," Math. Prog. 8, pp. 207-231.

Bus, J. (1977), "A Proposal for the Classification and Documentation of Test Problems in the Field of Nonlinear Programming," Proceedings Of NATO Advanced Study Institute on the Design and Implementation of Optimization Software.

Cline, A.K. (1977), "Two Subroutine Packages for the Efficient Updating of Matrix Factorizations," University of Texas at Austin, Department of Computer Science Report TR-68.

Cody, W.J. (1976), "Robustness in Mathematical Software", Proceedings of the Ninth Interface Symposium on Computer Science and Statistics, pp. 76-84, Prindle, Weber and Schmidt, Inc.

Colville, A.R. (1968), "A Comparative Study of Nonlinear Programming Codes," Report No. 320-2949, IBM New York Scientific Center.

Cottle, R.W. (1977), "Fundamentals of Quadratic Programming and Linear Complementarity," Systems Optimization Laboratory, Technical Report SOL 77-21, Department of Operations Research, Stanford University.

Crowder, H.P., Dembo, R.S. and Mulvey, J.M. (1977), "Guidelines for Reporting Computational Experiments in Mathematical Programming," Working Paper HBS 77-8 (rev.), Graduate School of Business, Harvard University.

Curtis, A.R., Powell, M.J.D. and Reid, J.K. (1974), "On the Estimation of Sparse Jacobian Matrices," J. Inst. Math. Applics. 13, pp. 117-119.

Dahl, O.J., Dijkstra, E.W. and Hoare, C.A.R. (1972), Structured Programming, Academic Press, New York.

Dantzig, G.B. (1948), "Programming in a Linear Structure," USAF, Washington, D.C.

Dantzig, G.B. (1968), "Large-Scale Linear Programming," in Mathematics of the Decision Sciences, G.B. Dantzig and A.F. Veinott, Jr., (eds.), pp. 77-92, A.M.S., Providence, R.I.

Dantzig, G.B. et al. (1973), "On the Need for a Systems Optimization Laboratory," Optimization Methods for Resource Allocation, English University Press, London.

Dantzig, G.B. (1977), "Linear Programming, Its Past and Its Future," Science Perspectives.

Dantzig, G.B., Eisenstat, S.C., Magnanti, T.L., Maier, S.F., McGrath, M.B., et al. (1970) "MPL -- Mathematical Programming Language -- Specification Manual," Computer Science Department, Stanford University, Technical Report STAN-CS-70-187.

Dantzig, G.B. and Parikh, S.C. (1977), "At the Interface of Modeling and Algorithms Research," Systems Optimization Laboratory Report SOL 77-29, Department of Operations Research, Stanford University.

Dantzig, G.B. and Perold, A. (1978), "A Basis Factorization Method for Block Triangular Linear Programs," Systems Optimization Laboratory Report SOL 78-7, Department of Operations Research, Stanford University.

Dantzig, G.B. and Wolfe, P. (1960), "Decomposition Principle for Linear Programs," Operations Research 8, No. 1, pp. 101-111.

Davidon, W.C. (1959), "Variable Metric Method for Minimization," Argonne National Laboratory, Report No. ANL-5990 (Rev.).

Davidon, W.C. (1975), "Optimally Conditioned Optimization Algorithms Without Line Searches," Math. Prog. 9, pp. 1-30.

Davidon, W.C. and Nazareth, L. (1977a), "OCOPTR -- A Derivative Free FORTRAN Implementation of Davidon's Optimally Conditioned Method," ANL-AMD Technical Memo. No. 303, Applied Mathematics Division, Argonne National Laboratory.

Davidon, W.C. and Nazareth, L. (1977b), "DRVOCR - A FORTRAN Implementation of Davidon's Optimally Conditioned Method," ANL-AMD Technical Memo. No. 306, Applied Mathematics Division, Argonne National Laboratory.

Dembo, R. and Mulvey, J.M. (1976), "On the Analysis and Comparison of Mathematical Programming Algorithms and Software," Proceedings of the Bicentennial Conference on Mathematical Programming, pp. 106-116, (to appear).

Dennis, J.E. (1976), "Non-linear Least Squares and Equations," A.E.R.E. Harwell, Computer Science and Systems Division, CSS 32.

Dennis, J.E., Gay, D.M. and Welsch, R.E. (1977), "An Adaptive Nonlinear Least-squares Algorithm," N.B.E.R. Working Paper No. 196, Cambridge, Massachusetts. (Also available as Technical Report No. 142, MIT Operations Research Center.)

Dixon, L.C.W. (1973), "Nonlinear Optimization: A Survey of the State of the Art," Numerical Optimization Center, Technical Report 42, The Hatfield Polytechnic.

Fiacco, A.V. and McCormick, G.P. (1964), "Computational Algorithm for the Sequential Unconstrained Minimization Technique for Non-linear Programming," Management Science 10, pp. 360-366.

Fiacco, A.V., and McCormick, G.P. (1966), "Extensions of SUMT for Non-linear Programming: Equality Constraints and Extrapolation," Management Science 12, pp. 816-829.

Fletcher, R. and Reeves, C.M. (1964), "Function Minimization by Conjugate Gradients," Comput. J. 7, pp. 149-154.

Fletcher, R. (1976), "Methods for Solving Nonlinearly Constrained Optimization Problems," Proceedings of York Conference on "State of the Art in Numerical Analysis," D. Jacobs, (ed.).

Fletcher, R. (1970), "A FORTRAN Subroutine for Quadratic Programming," A.E.R.E., Harwell Report No. R6370.

Fletcher, R. (1975), "An Ideal Penalty Function for Constrained Optimization," J. Inst. Math. Applics. 15, pp. 319-342.

Fletcher R. and Freeman, T.L. (1975), "A Modified Newton Method for Minimization," University of Dundee, Report No. 7.

Ford, L.R. and Fulkerson, R.E. (1954), "Maximal Flow Through a Network," The RAND Corporation, Paper P-605.

Ford, B. and Hague, S.T. (1974), "The Organization of Numerical Algorithms Libraries," in Software for Numerical Mathematics, D.J. Evans, (ed.), Academic Press, pp. 357-372.

Geoffrion, A.M. (1970), "Elements of Large Scale Mathematical Programming," Management Science, 16, No. 11, pp. 652-691.

Gill, P.E. and Murray, W. (1972), "Quasi-Newton Methods for Unconstrained Optimization," J. Inst. Math. Applics. 9, pp. 91-108.

Gill, P.E. and Murray, W. (1974), "Safeguarded Steplength Algorithms for Optimization Using Descent Methods," National Physical Laboratory, Report NAC 37.

Gill, P.E. and Murray, W. (1976a), "Linearly Constrained Problems Including Linear and Quadratic Programming," in Proceedings of York Conference on "State of the Art in Numerical Analysis," D. Jacobs, (ed.).

Gill, P.E. and Murray, W. (1976b), "Algorithms for the Solution of the Non-linear Least Squares Problem," National Physical Laboratory, Report NAC 71.

Gill, P.E. and Murray, W. (1977a), "Numerically Stable Methods of Quadratic Programming," National Physical Laboratory, Report NAC 78.

Gill, P.E. and Murray, W. (1977b), "A Brief Guide to the Numerical Optimization Software Library," NPL Algorithms Library Document, DNAC, National Physical Laboratory, England.

Gill, P.E., Murray, W. and Picken S.M. (1972), "The Implementation of Two Modified Newton Algorithms for Unconstrained Optimization," National Physical Laboratory, Report NAC 24.

Gill, P.E., Murray, W. and Pitfield, R.A. (1972), "The Implementation of Two Revised Quasi-Newton Algorithms for Unconstrained Optimization," National Physical Laboratory, Report NAC 11.

Gill, P.E., Golub, G.H., Murray, W. and Saunders, M.A. (1974), "Methods for Modifying Matrix Factorizations," Math. Comput. 28, pp. 505-535.

Gill, P.E., Murray, W., Picken, S.M. and Wright, M.H. (1977), "The Design and Structure of a FORTRAN Program Library for Optimization," Systems Optimization Laboratory, Technical Report SOL 77-7, Department of Operations Research, Stanford University.

Golub, G.H. and Pereyra, V. (1973), "The Differentiation of Pseudo-Inverse and Non-linear Least Squares Problems whose Variables Separate," SIAM J. Numer. Anal. 10, pp. 413-432.

Gomory, R.E. (1958), "Essentials of an Algorithm for Integer Solutions to Linear Programs," Bull. Amer. Math. Soc. 64, No. 5.

Guigou, G. (1971), "Presentation et Utilization du Code GREG," Note HI582/2, Electricite de France, 12 Av. de la Liberation -- 92 Clamart, France.

Han, S.P. (1975), "A Globally Convergent Method for Nonlinear Programming," Report No. 75-257, Department of Computer Science, Cornell University.

Han, S.P. (1976), "Superlinearly Convergent Variable Metric Algorithms for General Nonlinear Programming Problems," Math. Prog. 11, pp. 263-282.

Hestenes, M.R. (1969), "Multiplier and Gradient Methods," J.O.T.A. 4, pp. 303-320.

Hestenes, M.R. and Steifel, E. (1952), "Methods of Conjugate Gradients for Solving Linear Systems," J. Res. Nat. Bur. Stan. 49, pp. 409-436.

Hillstrom, K. (1976), "Optimization Routines in AMDLIB," ANL-AMD Technical Memo. No. 297, Applied Mathematics Division, Argonne National Laboratory.

Hillstrom, K. (1977), "A Simulation Test Approach to the Evaluation of Nonlinear Optimization Algorithms," ACM Trans. Math. Software 3, No. 4, pp. 305-315.

Himmelblau, D.M. (1972), Applied Nonlinear Programming, McGraw-Hill, New York.

Ho, J.K. (1974), "Nested Decomposition of Large Scale Linear Programs with the Staircase Structure," Systems Optimization Laboratory, Report 74-4, Department of Operations Research, Stanford University.

Jain, A. (1976), "The Solution of Nonlinear Programs Using the Generalized Reduced Gradient Method," Systems Optimization Laboratory, Report SOL 76-6, Department of Operations Research, Stanford University.

Kantorovitch, L.V. (1939), "Mathematical Methods in the Organization and Planning of Production," translated in Manag. Sci. 6, (1960), pp. 366-422.

Kerninger, B.W. and Plauger, P.J. (1974), The Elements of Programming Style, Bell Telephone Laboratories, Inc., Murray Hill, New Jersey.

Lasdon, L.S., Waren, A.D., Jain, A. and Ratner, M. (1976), "Design and Testing of a Generalized Reduced Gradient Code for Nonlinear Programming," Systems Optimization Laboratory, Report SOL 76-3, Department of Operations Research, Stanford University.

Lootsma, F.A. (1970), "Boundary Properties of Penalty Functions for Constrained Minimization," Phillips Res. Reports, Suppl. No. 3.

Lyness, J.N. and Kaganove, J.J. (1976), "Comments on the Nature of Automatic Quadrature Routines," ACM Trans. Math. Software 2, No. 1, pp. 65-81.

Madsen, R.E. (1974), "Users Manual for SEXOP (Subroutines for Experimental Optimization)," Release 4, Sloan School of Management, Massachusetts Institute of Technology.

Marquardt, D.W. (1963), "An Algorithm for Least Squares Estimation of Non-linear Parameters," SIAM J. 11, pp. 431-441.

Marwil, E.S. (1978), "Exploiting Sparsity in Newton-like Methods," Ph.D. Thesis, Report TR 78-335, Computer Science Department, Cornell University.

More, J.J. and Sorensen, D. (1977), "On the Use of Directions of Negative Curvature in a Modified Newton Method," ANL-AMD Technical Memo. 319, Applied Mathematics Division, Argonne National Laboratory.

More, J.J., Garbow, B. and Hillstrom, K.H. (1978), "Testing Unconstrained Optimization Software," ANL-AMD Technical Memo No. 324, Applied Mathematics Division, Argonne National Laboratory.

Motzkin, T.S. (1936), Doctoral Thesis, University of Zurich.

Muralidharan, R. and Jain, R.K. (1975), "MIN: An Interactive Educational Program for Function Minimization," Technical Report 658, Division of Engineering and Applied Physics, Harvard University.

Murray, W. (1969), "An Algorithm for Constrained Optimization," in Optimization, R. Fletcher, (ed.), Academic Press, New York and London.

Murray, W. and Wright, M.H. (1978), "Projected Lagrangian Methods Based on the Trajectories of Penalty and Barrier Functions," Systems Optimization Laboratory, Report SOL 78-23, Department of Operations Research, Stanford University.

Murtagh, B.A. and Saunders, M.A. (1977), "MINOS -- A Large Scale Nonlinear Programming System (for Problems with Linear Constraints), Users Guide," Systems Optimization Laboratory, Report SOL 77-9, Department of Operations Research, Stanford University.

Mylander, W.C., Holmes, R.L. and McCormick, G.P. (1971), "A Guide to SUMT-Version 4," Research Analysis Corporation Report RAC-P-63. Doc. AD-731391, National Technical Information Service, Springfield, Virginia.

Nash, J. (1975), Bibliography of Non-linear Least Squares (microfiche).

Nazareth, L. (1977), "Minkit -- An Optimization System," ANL-AMD Technical Memo. No. 305, Applied Mathematics Division, Argonne National Laboratory.

Nazareth, L. (1978), "Modules to aid the Implementation of LP Algorithms," Systems Optimization Laboratory Report SOL 78-28, Department of Operations Research, Stanford University.

Nazareth, L. (1975), "A Hybrid Least Squares Method," ANL-AMD Technical Memo. No. 254 (rev.), Applied Mathematics Division, Argonne National Laboratory, (to appear in A.C.M. Trans. on Math. Softw.).

Nazareth, L. and Nocedal, J. (1978), "A Study of Conjugate Gradient Methods," Systems Optimization Laboratory Report SOL 78-29, Department of Operations Research, Stanford University.

Nazareth, L. and Schlick, F. (1976), "The Evaluation of Unconstrained Optimization Routines," Proceedings of Bicentennial Conference on Mathematical Programming, N.B.S., Gaithersburg.

Nelder, J.A. and Mead, R. (1965), "A Simplex Method for Function Minimization," Comput. J. 7, pp. 308-313.

Orchard-Hays, W. (1954), "A Composite Simplex Algorithm -- II," The RAND Corporation, Research Memo. Rm-1275.

Orchard-Hays, W. (1977), "History of Mathematical Programming Systems," (manuscript).

Ortega, J.M. and Rheinboldt, W.C. (1970), Iterative Solution of Nonlinear Equations in Several Variables, Academic Press, New York-San Francisco-London.

Parkinson, J.M. and Hutchinson, D. (1971), "An Investigation into Variants of the Simplex Method," in Numerical Methods for Nonlinear Optimization, F.A. Lootsma, (ed.), Academic Press, London and New York, pp. 115-135.

Powell, M.J.D. (1964), "An Efficient Method for Finding the Minimum of a Function of Several Variables Without Calculating Derivatives," Comput. J. 7, pp. 155-162.

Powell, M.J.D. (1969), "A Method for Nonlinear Constraints in Minimization Problems," in Optimization, R. Fletcher, (ed.), Academic Press, London and New York, Chapter 19.

Powell, M.J.D. (1970), "A FORTRAN Subroutine for Solving Systems of Non-linear Algebraic Equations," in Numerical Methods for Nonlinear Algebraic Equations, P. Rabinowitz, (ed.), Gordon and Breach, pp. 115-161.

Powell, M.J.D. (1971a), A.E.R.E., Harwell Library Subroutine, VA05A.

Powell, M.J.D. (1971b), "On the Convergence of the Variable Metric Algorithm," J. Inst. Math. and Applics. 7, pp. 21-36.

Powell, M.J.D. (1975), "Restart Procedures for the Conjugate Gradient Method," A.E.R.E. Harwell, Computer Science and Systems Division, Report No. CSS 24.

Powell, M.J.D. (1976a), "Algorithms for Non-linear Constraints that use Lagrangian Functions," Presented at the Ninth International Symposium on Mathematical Programming.

Powell, M.J.D. (1976b), "A View of Unconstrained Optimization," in Optimization in Action, L.C.W. Dixon, (ed.), pp. 117-152, Academic Press, London-New York-San Francisco.

Powell, M.J.D. (1977), "A Fast Algorithm for Nonlinearly Constrained Optimization Calculations," presented at the 1977 Dundee Conference on Numerical Analysis.

Powell, M.J.D. and Toint, Ph.L. (1978), "On the Estimation of Sparse Hessian Matrices," Report DAMTP 78/NA1, University of Cambridge.

Reid, J.K. (1976), "FORTRAN Subroutines for Handling Sparse Linear Programming Bases," A.E.R.E. Harwell Report R 8269.

Rockafellar, R.T. (1973), "A Dual Approach to Solving Nonlinear Programming Problems by Unconstrained Optimization," Math. Prog. 5, pp. 354-373.

Rice, J.R. (1971), "The Challenge for Mathematical Software," in Mathematical Software, J.R. Rice, (ed.), Academic Press, New York and London, pp. 27-41.

Rosen, J.B. (1960), "The Gradient Projection Method for Non-linear Programming," Part I: Linear Constraints, SIAM J. Appl. Math. 8, pp. 181-217; Part II: Non-linear Constraints, J. Soc. Ind. Appl. 9, pp. 514-532.

Rosen, J.B. and Kreuser, J.L. (1971), "GPM/GPMNLC Extended Gradient Projection Method Nonlinear Programming Subroutines," Academic Computing Center, The University of Wisconsin, Madison, Wisconsin.

Rosen, J.B. and Wegner, S. (1975), "The GPM Nonlinear Programming Subroutine Package; Description and User Instructions," Technical Report 75-9, Computer Science Department, University of Minnesota.

Rosen, J.B. (1977), "Two Phase Algorithm for Nonlinear Constraint Problems," Technical Report 77-8, Computer Science Department, University of Minnesota.

Schubert, L.K. (1970), "Modification of a Quasi-Newton Method for Non-linear Equations with a Sparse Jacobian," Math. Comp. 24, pp. 27-30.

Shanno, D.F. (1977), "Conjugate Gradient Mqthods with Inexact Searches," MIS Technical Report No. 22, University of Arizona, Tucson.

Smith, B.T., Boyle, J.M. and Cody, W.J. (1974), "The NATS Approach to Quality Software," in Proceedings of IMA Conference on Software for Numerical Mathematics, J. Evans, (ed.), Academic Press, pp. 393-405.

Smith, B.T. (1976), "Fortran poisoning and Antidotes," In Portability of Mathematical Software, W. Cowell, (ed.), Springer Verlag Lecture Notes in Computer Science, Berlin-Heidelberg-New York, pp. 178-256.

Spendley, W. (1969), "Nonlinear Least Squares Using a Modified Simplex Minimization Method," in Optimization, R. Fletcher, (ed.), Academic Press, London and New York, pp. 259-270.

Staha, R.L. and Himmelblau, D.M. (1976), "Evaluation of Constrained Non-linear Programming Techniques," (manuscript), Department of Chemical Engineering, University of Texas, Austin.

Toint, Ph.L. (1977), "On Sparse and Symmetric Matrix Updating Subject to a Linear Equation," Department of App. Math. and Theoretical Physics, University of Cambridge, Report No. DAMTP 77/NA1.

Tomlin, J.A. (1976), "User's Guide for LCPL," Systems Optimization Laboratory, Technical Report SOL 76-16, Department of Operations Research, Stanford University.

Von Neumann J. and Morgenstern, O. (1944), Theory of Games and Economic Behavior, Princeton University Press, Princeton, New Jersey.

Wilkinson, J.H. (1960), "Rounding Errors in Algebraic Processes," Information Processing, pp. 44-53.

Wilson, R.B. (1963), "A Simplicial Algorithm for Convex Programming," Ph.D. Thesis, Graduate School of Business Administration, Harvard University.

Wolfe, P. (1959a), "The Simplex Method for Quadratic Programming," Econometrica, 27, pp. 382 et seq.

Wolfe, P. (1959b), "The Secant Method for Simultaneous Nonlinear Equations," Comm. ACM 2, pp. 12-13.

Wolfe, P. (1974), "A Method of Conjugate Subgradients for Minimizing Non-differentiable Functions," Report RC 4857 (#21613), IBM Thomas J. Watson Research Center, Yorktown Heights, New York, 10598.

Wolfe, P. (1975a), "Optimization: Concepts and Software," (manuscript).

Wolfe, P. (1975b), "Optimization," COSERS Panel on Numerical Computation, (draft manuscript).

Wright, M.H. (1978), "A Survey of Software for Nonlinearly Constrained Optimization," Systems Optimization Laboratory, Report SOL 78-4, Department of Operations Research, Stanford University.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER SOL 78-32	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) SOFTWARE FOR OPTIMIZATION		5. TYPE OF REPORT & PERIOD COVERED Technical Report
		6. PERFORMING ORG. REPORT NUMBER SOL 78-32
7. AUTHOR(s) L. Nazareth		8. CONTRACT OR GRANT NUMBER(s) N00014-75-C-0267
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Operations Research -- SOL Stanford University Stanford, CA 94305		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NR-047-143
11. CONTROLLING OFFICE NAME AND ADDRESS Operations Research Program -- ONR Department of the Navy 800 N. Quincy Street, Arlington, VA 22217		12. REPORT DATE December 1978 13. NUMBER OF PAGES 36
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

This document has been approved for public release and sale;
its distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Mathematical Software
Software Bibliography
Unconstrained Optimization
Optimization Software
Linear Programming
Constrained Optimization

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

SEE ATTACHED

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

SOL 78-32
SOFTWARE FOR OPTIMIZATION
L. Nazareth

Our aim in this paper is to provide the reader with:

a) Some feel for what quality software entails.
b) An overview of various aspects of optimization software.
c) Information on solution techniques and available software in the form of a decision tree.
d) An extensive bibliography so that the reader can further pursue specific topics of interest.

We concentrate upon linear programming, non-linear unconstrained optimization and related areas, and non-linear programming.

This paper is intended to supplement an earlier oral presentation at the Texas Conference on Mathematical Software entitled "State of Software for Optimization".